The Report Committee for Jeffrey Lynn Pinkston
Certifies that this is the approved version of the following report:

Designing a Consulting Services Architecture Model

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor:

Suzanne Barber

Thomas Graser

Designing a Consulting Services Architecture Model

by

Jeffrey Lynn Pinkston, B.S.

Report

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

May 2015

Dedication

The following report is an accumulation of four years of coursework, research, development and real-world implementation. Throughout this time period, I have relied on many co-workers, classmates and family for support and assistance in prioritizing my work. To each of you, whether it was to cover for me at work, explain in more detail a topic of discussion from a lecture or drive my kids to athletic practice, I offer my sincere thanks and appreciation. Without your help, this report would not have been possible. I must specifically thank my wife, Lisa, and daughters, Cassidy and Makenzie, for your patience and understanding as I travelled down this long road to graduation.

Acknowledgements

I would like to thank IBM employee Brad Schauf, IBM Executive Architect, for his guidance and source of information throughout the design, implementation and adoption of the ISSLMethod within IBM Software Services for Lotus (ISSL). The development of the ISSLMethod was the inspiration for the development of the model defined within this report, along with personal recommendations developed over the span of my coursework. Other resources that were instrumental in the development of the ISSLMethod include: Richard Gorzela, Hardy Groeger, David Pearson, Jonathan Pepin, Dimitry Radinsky, Mark Vardy, Hissan Waheed and Andreas Winter.

I would like to thank IBM employee Scott W. Ambler, Chief Methodologist for Agile and Lean, from the Rational Software Group for his work and presentation on the Introduction to Disciplined Agile Delivery (DAD). Scott's online resources were a tremendous benefit during my initial research for this topic.

I must also thank my fellow IBM Solution Architects, Jason Erickson and Terry Fouchey, who have backed me up on customer calls and sales meetings throughout the years as I have attended classes. Without this strong support structure in the workplace, I would never have had the drive to continue and complete this journey.

And last, I would like to thank my former manager, Larry Berthelsen, for providing the insight into issues that exist specifically within our services organization that should be addressed through the implementation of a services-specific methodology. His feedback allowed me to justify some of the same items that I had identified in the initial stages of my research.

Abstract

Designing a Consulting Services Architecture Model

Jeffrey Lynn Pinkston, MSE

The University of Texas at Austin, 2015

SUPERVISOR: Suzanne Barber

During my years of experience in the technology industry, it has become obvious that standard processes and methodologies within the engineering discipline are at a mature state. The realization though is that software engineering specifically lags behind. Most software engineering methodologies that I have studied focus on the mission of software development. It is this realization and the need for structure that led me to review existing methodologies used within my company's software services organization. The definition of what a successful software services methodology entails is rather limited. This report will provide a history of existing software engineering methodologies that I have studied, describe an initial services method that was being developed within my organization, develop a new model that addresses previous shortcomings and identify additional components required to further define a strong software services-oriented delivery methodology.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1 ONE METHOD DOES NOT FIT ALL

1.1 Overview

Defining one methodology to be used in all areas of computer engineering is not only difficult, but not feasible. Each and every discipline has specific requirements that differentiate it from others. Working for a large technology company has advantages and disadvantages, as it offers opportunities to review what methodologies are being used within each group. The various groups within the company implement many different software engineering methodologies and the frameworks differ by the main responsibility of each group. Most projects that are very broad in nature and have a duration measured in years utilize the Unified Method Framework (UMF). For this reason, UMF is seen as too complex to work for these types of projects.

The Waterfall method has long been the post facto standard whether for software development or services solution implementations. Currently, the popular concept is the use of the Agile method, as well as the Unified Process (UP), SCRUM and Extreme Programming (XP) methods to provide solutions. The focus of this paper will be on the development of a new model based on multiple existing software engineering methodologies but modified specifically for the design and implementation of systems from a services organization viewpoint. The model developed in this report is labeled as the Consulting Services Architecture Model (C-SAM).

1.2 Introduction

The development of a method utilized specifically for a service organization must contend with the fact that each and every project can be handled differently depending on the situation. On a broader scale in which an organization provides services to multiple customers, there can be an even larger number of scenarios. Multiple groups may share common goals and interests, but there are always various groups that require special treatment. There are many different software engineering lifecycle models to choose from, such as Waterfall, Spiral, and Prototyping [see Figure 1.1]. Within a technology company, it is evident that a "one size fits all" way of thinking does not work. There are very specific differences between the same models and methodologies used by different software development teams and those used to deliver a services solution for a customer. To understand the differences and requirements for developing a service model, we must understand the foundation and complexities of the methodologies and frameworks currently available. It is this understanding of what exists today that lead to the research and development of the model presented in this report.

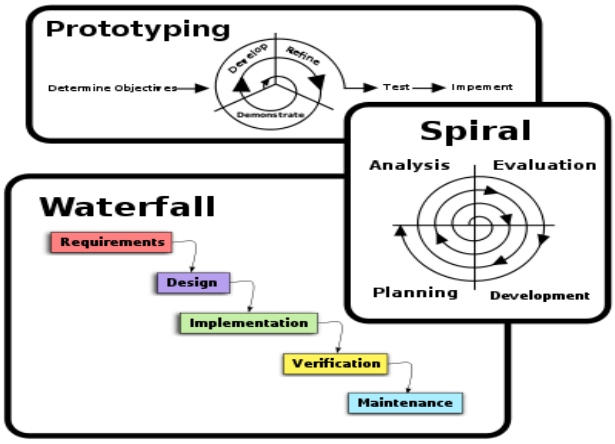

Figure 1.1. Example models [6]

1.3 Frameworks

The following concepts are provided to gain an understanding of what each provides or lacks when designing and implementing a methodology framework within a service organization. Many features found in these examples provide a foundation for components that are required and implemented within a service model.

1.3.1 Unified Method Framework (UMF)

Although different organizations deliver different services to their clients, they often refer to or produce similar work products. Prior to the IBM Unified Method Framework (UMF) [7], most methods were task-based and each line of business independent from one another, making it hard to deliver end-to-end solution-defined work products. UMF contains a set of work products that provide a "common language" for all practitioners. These work products provide the basic

building blocks for constructing different delivery processes and capability patterns to perform specific types of project.

UMF provides the guidance on what common work products to create and how to create these within activities, tasks and roles [see Figure 1.2]. The work products defined by UMF are a critical component for the development of the new services delivery model contained in this report.

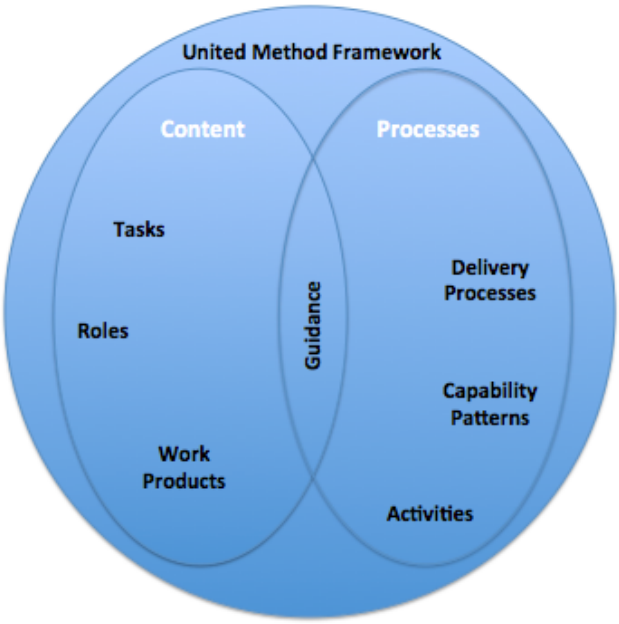

Figure 1.2. Unified Method Framework [7]

1.3.2 Rational Unified Process (UP)

The IBM Rational Unified Process (UP) [4] is an approach that is used mainly to develop software products. It contains information about:

- the type of work needed to develop software (tasks)
- the sets of responsibilities we assign to people (roles)

- the items to produce (work products)
- the assistance in performing this work (guidance).

This process is similar to the previously mentioned Unified Framework. The following key concepts are important to UP:

- Iteration phases – breaks the development process into smaller sections, allowing for simplicity within each section for development, support, and maintenance.
- Architecture-driven – this concept provides the environment necessary to execute the solution to provide feedback for the development prior to delivering the solution into production.
- Use Case driven – allows the process to take the solution through to implementation providing solutions to business use cases and solving specific functionality.

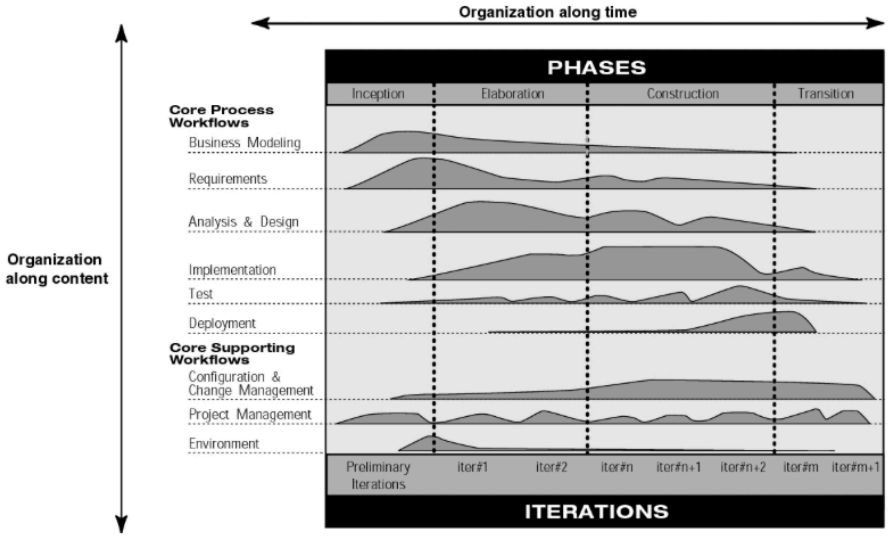

Figure 1.3. Nine cores of IBM Rational Unified Process [5]

1.3.3 EXTREME PROGRAMMING (XP)

Extreme Programming (XP) is a framework built on the Agile process. The main focus of XP is the goal of customer satisfaction, not the development of work products specifically. Similar to UP, XP is iterative in nature, providing constant feedback and multiple phases of designing, developing and implementing a customer solution. Relating this to a service project is very difficult as there are many basic components that typically must be completed prior to moving to the next step. But implementing an iterative model, based on feedback from a customer throughout the duration is an important part of a successful project.

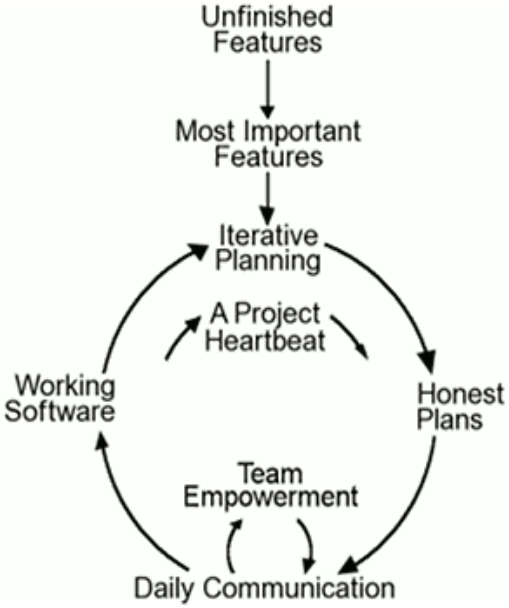

Figure 1.4. Extreme Programming (XP) concept [8]

1.3.4 SCRUM

The Scrum framework is another approach to software development using the Agile methodology. The main focus of Scrum is the flexibility to adapt to the ever-changing requirements of a software development project. Scrum divides a project into sprints, or short periods of duration, in which to develop and implement specific functionality based on real world experiences and estimations. Most services projects require a stable set of requirements from the beginning to define the scope of the project. There must be a process to provide changes in scope based on new requirements found during project implementation.

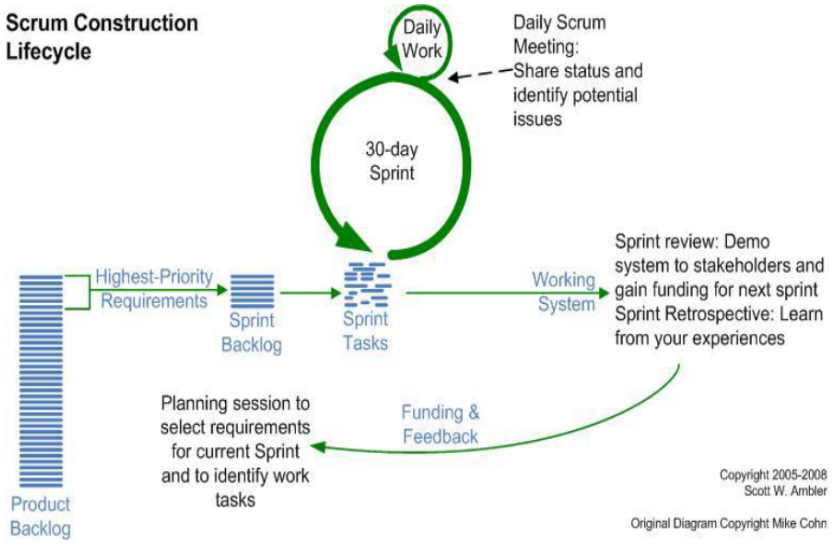

Figure 1.5. Scrum lifecycle [1]

1.4 Observations

The preceding methodologies and frameworks were researched and discussed with various co-workers at IBM to define a framework and develop a model specific to a service organization. The information gathered concluded that there are many methodologies available, but at this point, none that were specific to software services organizations focused on providing customer solutions outside of the standard software development methods.

These discussions provided an insight as to one specific project currently involved in the development of new methodologies within IBM. The Disciplined Agile Delivery (DAD) is a hybrid approach using many of the concepts previously described as a solution framework. By reviewing this approach and studying the strengths and weaknesses, a new services model named Consulting Services Architecture Model (C-SAM) was developed.

CHAPTER 2 IDENTIFYING A NEED

2.1 Overview

From an approach standpoint, as with most software services organizations, there is a basic conflict between mission and practice. There are standards that are designed to drive solutions, but the ultimate goal of organizations is profitability.

This scenario often leads a services organization to take direction from departments such as the Finance Department or even Systems Operations groups. These groups do not understand what is technically required to compete in today's market. This specific issue is compounded even more when dealing with these types of departments funding project work. Many projects funded by Finance or Real Estate Departments have failed because they did not initially include the IT department.

Many technical organizations, not specifically services, have issues with accumulating data from their project experience. Being able to access the actual information from this data would be helpful. The lack of accumulating data to measure the success of a project is a fundamental flaw in the implementation of the methodology.

Any of the preceding scenarios would lead to inefficiencies in implementing a solution for their customers. Inefficiency drives up the cost of these solutions, which are then passed on to the customer. Once this overhead reaches a certain

point, the organization has to design a higher pricing model, which translates into a poor competitive position and loss of business.

2.2 Introduction

The concerns previously listed are the basis for developing a standard model for services organizations to follow. Following a specifically designed services model allows the organization to build a solid foundation from the beginning using well-documented and proven processes, procedures and techniques. The Disciplined Agile Delivery framework is an example of the next step in developing a new model based on the methodologies and frameworks previously listed in this report.

2.2.1 Disciplined Agile Delivery (DAD)

The Disciplined Agile Delivery (DAD) concept is designed as a hybrid framework using many of the previous models as a source for adopting the best practices and philosophies of several methodologies [see Figure 2.1].

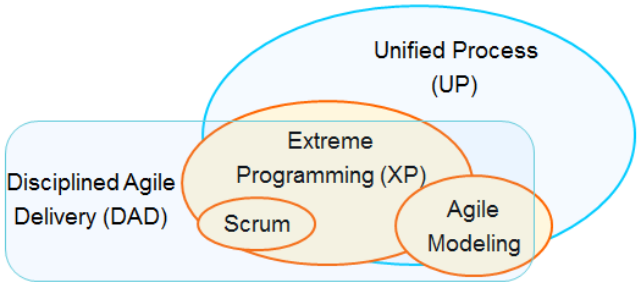

Figure 2.1. Disciplined Agile Delivery concept [1]

The DAD process framework adopts the specific components and strategies from the following methods:

1. Scrum - The focus of Scrum is on project leadership and some aspects of requirements management. DAD uses many ideas from Scrum [1], such as:
    - working from a stack of work items in priority order
    - having a Product Owner responsible for representing stakeholders
    - producing a potentially consumable solution every iteration
2. Extreme Programming (XP) - XP is an important source of development practices for DAD, including, but not limited to:
    - continuous integration (CI)
    - refactoring
    - test-driven development (TDD)
    - collective ownership
3. Agile Modeling (AM) - AM is the source for DAD's modeling and documentation practices. This includes:
    - requirements envisioning
    - architecture envisioning
    - iteration modeling
    - continuous documentation
    - just-in-time (JIT) model storming

4. Unified Process (UP) - DAD adopts many of its governance strategies from agile instantiations of the UP, including OpenUP and Agile Unified Process (AUP). [1] In particular, this includes strategies such as having lightweight milestones and explicit phases. It also draws from the UP focus on the importance of proving out the architecture in the early iterations and reducing all types of risk early in the lifecycle.
5. Agile Data (AD) - AD is a source of agile database practices, such as:
    - database refactoring
    - database testing
    - agile data modeling
6. Kanban - DAD adopts two critical concepts from Kanban, which is a lean framework model:
    - limiting work in progress
    - visualizing work

2.2.2 CHARACTERISTICS OF DAD

The DAD process framework provides a customer delivery solution built on a people-first, learning-oriented approach. It has a risk/value lifecycle, is scalable, is goal-driven, and is enterprise aware. There are several important characteristics of DAD that are critical when deciding on a service methodology.

People first

Within DAD, there are primary and secondary roles identified by each team. For identification and review in developing a new model for services, the focus will be on the primary roles of DAD:

- Stakeholders – all users affected or who affect the system.
- Team Lead – responsible for the success of the project and employing the process to build the solution.
- Product Owner – defines and promotes the vision, goals and capabilities of the solution.
- Agile Team member – members of the delivery team.
- Architecture owner – understands the architectural direction of the solution.

Learning oriented

As one of the key characteristics of DAD, a learning environment was identified as critical for most effective services organizations. There are three specific aspects of the learning oriented characteristic:

- Domain learning – identifying what the stakeholders need and how services will help them achieve what they need.
- Process improvement – being able to track improvements and changes needed at the end of each iteration.
- Technical learning – understanding how to work effectively with tools and technology that is available to the team.

Hybrid and Agile

DAD is a process framework that can be modified to meet the needs of each situation. Agile methods such as Scrum and XP include concepts popularized by UP. UP has evolved to address many of the new concepts popularized in agile methods.

Goal-driven

Projects evolve, and the work emphasis changes throughout the lifecycle. DAD divides the project into phases with milestones to ensure focus on the right areas. Some of the areas include initial visioning, architectural modeling, risk management, and deployment planning. This model differs from mainstream agile methods, which typically focus on the construction aspects of the lifecycle.

Simply indicating goals is of little value. A goals-driven approach provides guidance for service delivery teams but allows the flexibility for these teams to customize the process to address the issues specific to their situation.

Risk and value-driven

DAD is an evolutionary (iterative and incremental) approach that regularly produces high-quality solutions in a cost-effective and timely manner. It is performed in a highly collaborative, disciplined, and self-organizing manner within an appropriate governance framework, with active stakeholder participation to ensure that the team understands and addresses the changing needs of its stakeholders. [1]

Enterprise aware

DAD teams work internally within an organization's enterprise environment and try to take advantage of the opportunities presented to them. This includes working closely with:

- technical architects and engineers to leverage and enhance the existing and future technical infrastructure
- business architects and portfolio managers to fit into the overall business
- senior managers to govern the various teams appropriately
- data administrators to access and improve existing data sources
- IT support resources to understand and follow enterprise IT guidance.

In other words, DAD teams adopt a mindset of designing and developing for the entire enterprise, as the foundation to build a service model.

2.2.3 SUMMARY OF DAD

The Disciplined Agile Delivery framework is a very defined and detailed process that contains the flexibility needed for almost any type of services engagement required in the environments for this research. The scalability that is inherent within DAD provides ability for software services organizations to use the fundamental concepts of DAD for almost any size project. A software development team focuses on the functionality and development of their product. A software services team focuses on the customer, their business and providing a solution that meets their needs.

CHAPTER 3 DRIVING DELIVERY SERVICES WITH C-SAM

3.1 Overview

As a next step in building a new model for service organizations, it is very important to apply a common project methodology to ensure the same high delivery standard across all engagements. This step requires having a common way of defining an engagement, delivering it based on such definition, and creating appropriate completion documentation. It is also important to store and harvest intellectual capital (IC) to be re-used and refined in future projects. These steps drive the development of the Consulting Services Architecture Model (C-SAM).

C-SAM leverages IBM methodology standards such as Unified Method Framework (UMF) and the Disciplined Agile Delivery (DAD) framework to define a model for services specific engagement types based on relevant work products. C-SAM will provide the ability to implement a strong foundation using a minimalistic approach to a service methodology.

Through the combination of output centric models and out-of-the-box guidance, templates, and intellectual capital, C-SAM provides a minimal but sufficient methodology framework. Minimal but sufficient enables practitioners of C-SAM to run projects based on proven delivery models while allowing experts to focus on the technical delivery work. While relying on UMF and DAD in terms of structuring and describing an engagement model, C-SAM provides simplified tooling to lower the barrier of using it in small to medium size projects.

One inherent aspect of the C-SAM is the continuous focus on enhancing the existing model as well as extending through additional models for typical engagements. C-SAM relies on building a strong foundation of work products for solutions to be implemented and built upon these work products using feedback processes at the completion of each project.

3.2 Building the C-SAM Model

An easy-to-use project methodology provides value by increasing customer satisfaction through more consistent delivery excellence in both scoping and delivering services projects. Improved delivery service and higher customer satisfaction are important steps towards implementing a successful service methodology.

Although project methodologies can provide direct benefits to the business, they have often been difficult to apply. Methods such as UMF provide a very powerful and generic framework as well as a vast number of predefined delivery processes. However, the power and size of these models and related tools has left many practitioners feeling that they are too complex, too generic or require too much overhead to apply in a cost-effective way. Therefore, the development of C-SAM results in a consistent and simplified approach for structuring the delivery of the service engagement itself.

3.2.1 GOAL OF C-SAM

The goal of C-SAM is to define a "minimal but sufficient" method approach and tooling that a delivery resource will want to use. All team members should feel confident in applying a consistent project methodology. Even without a methodology subject matter expert on a project, C-SAM provides a beneficial approach that is lightweight. C-SAM also provides a high degree of out-of-the-box value through directly reusable work products while still allowing the model to adapt to specific engagement requirements. In other words, the goal of C-SAM is to:

- Standardize on a small set of well-defined project artifacts for each engagement type
- Collect and maintain technology-specific guidance, reusable IC, and templates
- Use a “Work Product Based Approach" to build up a reliable stack of reusable work products over time
- Work smart, not hard – take advantage of common standards and follow a predictable path from sales to delivery

One important tool used within the C-SAM method to guide the resources to goal-driven success is the implementation of a strong project plan. Multiple project plan templates are developed based on the service or specific product type of implementation that is common for a service organization. A project plan template for each specific project type is the basic foundational tool for use with new project requests. Over time, each specific project plan template is modified based on the

results of previous engagements that allow for better estimation of effort in future projects.

Activity 3: Build	**35.5 days**	**438 hrs**
Production Environment Installation and Configuration	**35.5 days**	**422 hrs**
Primary Environment	**35.5 days**	**422 hrs**
Staging Files / Install WAS	3 days	48 hrs
Install RDBMS (2 days per server) (2)	4 days	32 hrs
Install Product Platform Server (1.5 days per server)	1.5 days	24 hrs
Install System Console (.5 days per server)	0.5 days	4 hrs
Install Community Services (.3 days per server) (9)	3 days	48 hrs
Install Community Services MultiPlexor (.2 days per server) (5)	1 day	16 hrs
Install Meeting Server - Base (.25 days per server) (4)	1 day	8 hrs
Install Meeting Server - Capturer/Renderer/Conversion (.25 days per server)(4)	1 day	8 hrs
Install Meeting Server - Proxy - Intranet (.5 days per server) (3)	1.5 days	12 hrs
Install Meeting Server - Proxy - DMZ (.5 days per server) (2)	1 day	8 hrs
Install Proxy Server - Intranet (.5 days per server) (9)	4.5 days	36 hrs
Install Proxy Server - DMZ (.5 days per server) (2)	1 day	8 hrs
Install Gateway Server (1 day per server) (2)	2 days	16 hrs
Install SIP/XMPP Proxy (.5 days per server) (2)	1 day	8 hrs
Install Advanced Server (1 day per server)(2)	2 days	16 hrs
Install AV - Video MCU (.5 days per server) (2)	1 day	8 hrs
Install AV - Video Manager (1 day per server) (3)	3 days	24 hrs
Install AV - Proxy Registar (.5 days per server) (7)	3.5 days	28 hrs
Install AV - Proxy Registar (WAS Proxy) (.5 days per server) (3)	1.5 days	12 hrs
Install AV - Conference Focus Manager (.5 days per server) (2)	1 day	8 hrs
Install AV - Conference Manager (WAS Proxy) (.5 days per server) (2)	1 day	8 hrs
Install AV - Bandwidth Manager (.5 servers per day) (2)	1 day	8 hrs
Install AV - TURN Server (.75 days per server) (3)	2.25 days	18 hrs
Install AV - WebSphere SIP Edge Server (1 day per server) (2)	2 days	16 hrs
Client Configuration	**1 day**	**8 hrs**
Build client deployment packages (up to 2)	0.5 days	4 hrs
Prepare and configure for client deployment	0.5 days	4 hrs
Environment Testing	**1 day**	**8 hrs**
Migrate DNS to new environment and test	1 day	8 hrs

Figure 3.1. Sample Build Activity

Figure 3.1 provides just one example of how the how a specific project plan is used to provide quick and valid estimates based on previous work of similar type. This example shows all of the components for the implementation of an instant messaging and web conferencing solution. Each task provides the estimated effort for the build of each component. The last number of each task is the number of installation instances of each task. No number listed implies a single installation instance of this task. The final step for calculating the effort for each task is simply to

multiply the recommended duration by the number of instances to get the effort for each task. Components not included in the project are simply disregarded by inserting zero days or hours in the duration column. This process is followed throughout each of the stages of C-SAM as described in section 3.2.4.

3.2.2 KEY ASPECTS OF C-SAM

Some of the key aspects of C-SAM are:

- Engagement Models define work-product and output centric views only. C-SAM focuses on creation rather than when or in what sequence to create something (as in a process-centric view).
- C-SAM includes a set of mandatory work products (Core Model) and a set of optional work products. Optional work products are identified through the use of a questionnaire, which needs to be answered based on project- specific requirements. This results in an adopted model reflecting the list of required work products for a specific project or engagement type.
- Work products in C-SAM are directly reused from UMF whenever possible and can be modified to fit a service engagement.
- In addition to the core model and questionnaire, service packages containing guidance documents, templates and reusable intellectual capital (IC) specific to a certain technology are provided as part of an engagement model.

3.2.3 ENGAGEMENT TYPES OF C-SAM

Service organizations are involved in multiple internal and customer-related engagement types. Infrastructure builds and upgrades, as well as custom integrated solutions, are the major categories of engagements for a service project. C-SAM characterizes each project within the following types of engagements and customer projects as part of the service method:

- Custom Application Development – the standard development model for a customized application installation or development project.
- Install/Setup/Configure – the standard deployment model for the initial implementation of a solution and is infrastructure oriented.
- Upgrade / Migration – the installation of a new version of an existing system software or the installation of a new system involving the movement and conversion of source data to a target system format.
- System Health Check / Assessment – the standard current environment review model; can be infrastructure or application-oriented.

It is important to the success of C-SAM that additional engagement types are added and refined to capture important aspects of each project for future use. As previously stated, C-SAM is focused on representing a simplified model based on UMF and DAD for small to medium size projects not requiring the complexity of a full UMF project. Very large-scale customer projects should be looking at UMF in its full breadth since the pre-tailoring done for C-SAM may have eliminated relevant

work products for the sake of simplicity. Figure 3.1 illustrates the minimalistic approach of the C-SAM engagement model. The basic idea is to divide the model into resources and work products.

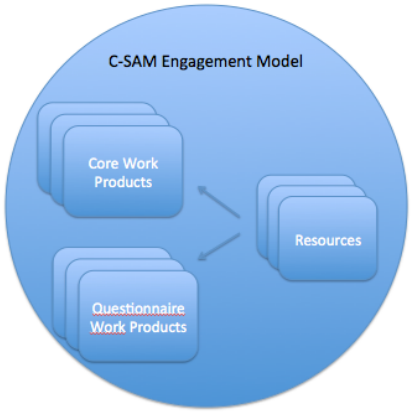

Figure 3.2. Consulting Services Architecture Model

3.2.4 STAGES OF C-SAM

The C-SAM method follows a Waterfall type framework for the stages of delivery. Each stage builds upon the previous. Since one of the key aspects of C-SAM is to divide project responsibilities between the resources and work products, multiple stages can be run in parallel, based on the assigned resource for each delivery stream. The only requirement is that previous foundational tasks are completed before initiating the next stage. For example, multiple build streams can exist in the Build and Test stage for the initial installation of multiple software products prior to any integration tasks between the systems. The stages for C-SAM are:

- Foundation Discovery – The initial stage of a service engagement containing a project kick-off meeting, a requirements review, and a current environment assessment.
- Solution Design Planning – Implementation plans and resource schedules are created based on the definition of the business and technical requirements and the status of the technical infrastructure.
- Build and Test – This stage entails the hands-on work for implementing the plans and confirming the completion of the systems defined in the Solution Design Planning stage.
- Implement – All solutions require the deployment of a user community. This stage fulfills the system access requirements by the designated user community.
- Enablement, Maintenance, and Support – This stage is the final phase of the solution delivery. The tasks in this stage include training of the user community, including users, administrators and support personnel. Maintenance and Support provide daily monitoring and response to system issues.

3.2.5 MEASURING C-SAM

The goal of any service organization is to improve the delivery of solutions to their customers, whether they are internal or external. One measurement of the success of a consulting organization is to use the Capability Maturity Model Integration (CMMI) [2]. CMMI is a process improvement framework for appraising the maturity of the services (or development) organization. CMMI levels include:

- Level 0 – Chaotic
- Level 1 – Heroic
- Level 2 – Managed
- Level 3 – Defined
- Level 4 – Quantitatively Managed
- Level 5 – Optimized

The organization researched for this report has achieved a Level 1 maturity, meaning that most projects performed in an ad-hoc, heroic manner. Project achievement and requirements success based on personal resource experience and heroism. The development of a new service method specific to a service organization suggests the need and drive to help improve the CMMI maturity level. The development of C-SAM is perceived to provide a standard methodology to meet a higher level CMMI maturity level.

3.2.6 APPLYING C-SAM

The following is a conceptual overview of applying C-SAM:

Start with Core Work Product Set

Through the C-SAM engagement model, the practitioner is given a minimal core set of work products mandatory for the chosen engagement type [see Section 3.2.7]. These work products are based on UMF, but chosen in the context of small to medium engagement sizes. Basic domains, such as architecture or project

management, isolate their work products of interest. The result is that practitioners quickly have a small but sufficient set of work products to give them initial focus.

Leverage optional project specific Work Products

The practitioner utilizing C-SAM can also identify additional work products specific to an engagement. Instead of going through a long list of work products, the practitioner can start with a specific engagement type. The practitioner can then select the required Work Products from a recommended list based on prior experiences with the service organization. As with the core work products, the optional work products are grouped by basic engagement type. The practitioners select project specific work products based on project requirements.

Leverage Resources to create Work Product instances

After the selection of the core set and required optional work products, additional resources will help the practitioner with the creation of the necessary work product materials for an engagement. Resources can be guidance documents like checklists or templates and other intellectual capital (IC). Resources are based on UMF guidance artifacts, but tailored to project needs.

FeedForward and FeedBack

The C-SAM approach includes a lightweight approach to feeding back into the method for continued improvement. As previously stated, C-SAM is work-output centric. Capturing feedback and measuring the success of a project is fairly difficult, resulting mainly in a yes or no response when questioning success. Therefore,

categorizing projects and archiving work products are one feedback exercise to implement improvements to the model within the specific engagement type. The requirement for capturing this information led to the development of the FeedForward/FeedBack process.

Sharing information between a sales organization and the technical delivery team and within groups in the delivery team is a very important. This aspect provides data to and captures data from the resources that successfully impact projects using C-SAM. Initially, the sales team must be able to forward information regarding the configuration of the solution the customer has requested. It is imperative that technical resources be part of the sales team during this initial development of the solution. The technical resources may not be part of the actual implementation; therefore, there must be a structured way to get this information to the resources that deliver the project. At a minimum, an initial checkpoint should be scheduled to share all current information among team members.

One or more FeedBack checkpoint should occur once the project has begun. Checkpoints provide the opportunity to capture ongoing information as to the status of the project. Short, periodic FeedBack checkpoints are recommended over long, end-of-project wrap-up meetings. The FeedBack checkpoints provide the opportunity to monitor the current project and update specific tasks in the project plan template for this project type in the scoping of future engagements.

One example of how the FeedBack process is important is with the release of a new software product version. Based on previous experiences with the same product, an estimated duration for the installation is calculated based on the total of all required tasks. It is common that a new release of software causes great changes in the effort required for installation. Changes can work both in a positive and negative direction. The product may become more difficult to implement due to multiple features and functionality added. Alternately, the product's installation script may automate many components better than previous versions and allow for easier, faster installation. Capturing FeedBack whether from an actual project implementation or a test environment installation provides the biggest impact on the generation of project estimates.

Because C-SAM is document-based, an IBM Domino [3] application was selected as the database tool for recording and storing work products. IBM Domino is an application platform and collaboration system specializing in document management and workflow functionality. Therefore, each project will store all collateral within a single engagement model database on an IBM Domino server. Figure 3.3 illustrates the engagement database for the Consulting Services Architecture Model.

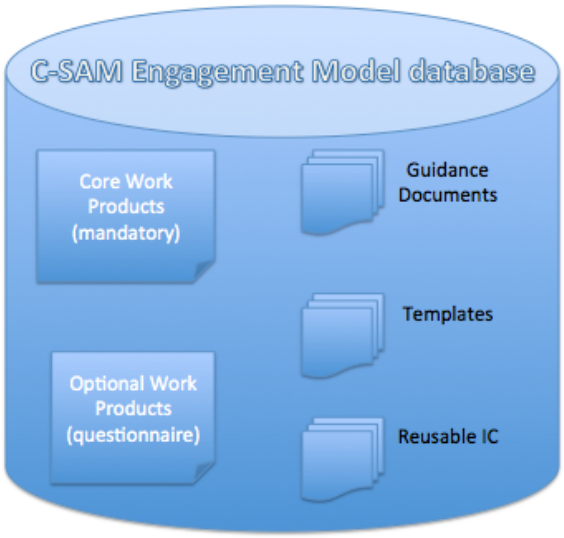

Figure 3.3. Consulting Services Architecture Model database

3.2.7 TECHNICAL WORK PRODUCTS

The technical work products are one of the most important fundamental aspects of C-SAM. Delivering a high-quality standard solution utilizing documentation created from best practices and experience is the goal of implementing C-SAM. As previously defined, C-SAM divides work products into Core and Optional categories. These work products are provided in document format and developed during the beginning phases of a C-SAM engagement. Each work product targets business stakeholders, as justification for all decisions and modifications made throughout the project. Certain documents, such as the Requirements Matrix, are developed and finalized prior to the development of other work products and delivery of the project. The delivery team continuously updates the remaining Core work products throughout the project.

Core Products

- Project Charter – describes the project objectives. The purpose of a charter document is to define the reasons for undertaking the project. The charter describes the objectives and constraints of the project and identifying the main stakeholders of the project.
- Environment Foundation – provides an assessment of the current environment in preparation for the engagement type to implement. The Environment Foundation provides the foundational baseline in which the project will build to define the functional and technical quality of the new environment. Included in this work product are the resource roles and responsibilities of the organization and the existing IT strategy policies and procedures.
- Requirements Matrix – details the functionality from a technical and business viewpoint. This matrix is generated from initial discussions and workshop meetings between the stakeholders and the project delivery team to provide requirements and identify gaps in fulfilling a successful project. The typical areas of concern for this work product are:
    - Availability
    - Backup and Recovery
    - Capacity Estimation and Planning
    - Configuration Management

    - Disaster Recovery
    - Extensibility and Flexibility
    - Failure Management
    - Performance
    - Scalability
    - Security
    - Service Level Agreements
    - System Management and Support
- Architecture Overview – diagrams the overall architectural vision of the project. This overview assists with the understanding of the future direction and helps the management decision-making process. The Architecture Overview document contains the following sections:
    - Enterprise Level
        - Description of the overall enterprise level architecture
        - Pictorial diagram describing each component in the enterprise level architecture
        - Key conceptual description of the components defined in the enterprise level architectural diagram
    - System Level
        - Description of the system level architecture

        - Pictorial diagram describing each component in the system-level architecture illustrating the features and functionality of the proposed solution
        - Detailed description of each component's feature and functionality within the proposed solution
        - Key conceptual description of the components defined in the system level architectural diagram, including:
            - Range of delivery mechanisms supported
            - Separations of functions in the proposed architecture
            - Architectural model, such as three-tier, four-tier, etc…
            - Definition of each hardware component feature
            - Required access to legacy systems
- Architectural Decisions – documents all key architecture decisions and the rationale behind each decision. This document ensures that there is a single source of consistent decisions being communicated to the project team. Figure 3.4 provides an example of the information required by each key decision made based on the project requirements.

Architectural Decision 001		Topic	Topic of interest
		ID	A unique identifier
Subject Area	Overall area of concern		
Architectural Decision	Summary of the decision made, indicating what the decision is.		
Issue or Problem Statement	A short description of the problem, what is being decided		
Assumptions	What is the context of the problem? Can you identify any constraints on the solution?		
Motivation	Why this decision is important? Are business factors impacting this decision?		
Alternatives	Are there any alternatives to the decision proposed?		
Decision	What is the final decision? Does it relate to other work products?		
Justification	Why was this decision made? Provide a list of compliance to architecture principles and explanations of deviations from compliance.		
Implications	What impact the decision will have?		
Derived requirements	What requirements are generated by this decision?		
Related Decisions	What other decisions are related to this decision?		

Figure 3.4. Example Architectural Decisions work product

- Implementation Plan – defines all activities required for the project. This work product is one of the most important used in conjunction with the project plan for the implementation of the solution. The project needs to cover the number of environments being installed and the skill level of each resource required to the number of resources needed to complete the project.
- Build Procedures – describes the executable procedures required to generate a copy of the installed system. This work product serves two specific purposes: (1) to document the specific settings during the build of the current environment and (2) provide reviewable material for future troubleshooting and system duplication exercises.

Optional

- Configuration Parameters – sets the selection of values and options implemented in the system. This work product documents the rules and

standards for settings within the system, as well as the actual values that were input during the installation of the system.

- Component Design – provides a functional view of the system. Components include the structure, modularity and behavior of each piece of the solution.
- Operational Design – describes the required operational capabilities of the installed system. This work product is a design review, isolating problems that occur after implementation. This document is the foundation of a training guide for the administration and support teams.

3.2.8 PROJECT MANAGEMENT DOCUMENTS

In addition to the technical work projects, C-SAM requires specific project management documentation to assist in minimizing delays and increasing the success of the implementation. C-SAM requires the following work products:

- Work Breakdown Structure – defines the task schedule and assigned resources for the project. Sometimes labeled as the Project Plan, this product is important in keeping the overall project on track and fully staffed to handle all assigned tasks for the project.
- Project Status Report – provides management with a current written assessment of the state of the project. This report delivers a weekly status and contains the health of the project, project accomplishments, current risks and a status of the resources assigned.

- Risk Definition – defines any risks that may impact the project. The purpose of this is to plan for any risks identified and communicate these risks to the project stakeholders.
- Issue Log – describes issues encountered during the project. This log provides information regarding how risks are being managed throughout the project.
- Communications Plan – provides information to the user community in preparation for use of the new system. Communications plans are important to keep users aware of changes that are coming, whether it is new functionality or a completely new system that is being provided. Proper communication decreases the amount of support required for implementation.

3.3 Future of C-SAM

Each technical resource responsible for the design, development and delivery of a solution impacts the success of the C-SAM model. The model is updated based on feedback from the technical delivery team. As with anything new, there is a learning curve by those that will deploy a project using the methodology and tools provided to support C-SAM. Over time, as the new model and support tools begin to mature, C-SAM may be seen as the standard methodology for software services

organizations. Successful implementations can also improve the current CMMI maturity level as the approach provides:

- Consistent, minimum but sufficient methodology framework to provide structure and guidance on how to deliver projects (CMMI Level 2 – Managed).
- Adoption or tailored models to specific project needs and feed enhancements, as well as intellectual capital back into the model (CMMI Level 3 – Defined).

Acceptance by the technical community impacts the ability of C-SAM to reach higher CMMI levels over time.

CHAPTER 4 SUMMARY

By understanding the current models, methodologies and frameworks available in software engineering, the development of new models, such as C-SAM, advance specific areas in this discipline. Software service organizations lack the maturity to have standard methodologies built that pertain to services type engagements. But learning the fundamental methods, such as Unified Method Framework, to build a services-oriented methodology is a good start for a foundation. Disciplined Agile Delivery method is one of the latest projects at IBM used by the organization. This method is a cross model of several accepted standards such as Extreme Programming, Agile modeling, and Scrum. C-SAM is a new model based on the combination of these existing standards and best practices resulting from many years of services experience.

C-SAM utilizes the development of specific work products by appropriately skilled resources to implement a technical solution based on a standard model for multiple engagement types. Re-using and updating document collateral over time, utilizing FeedForward and FeedBack checkpoints, will help increase the validity of the C-SAM method, creating a standard to use for future technical engagements. The final goal of C-SAM is to continue to develop into a "minimal but sufficient" model for software service engagements.

REFERENCES

The following references, although not specifically used for documentation within this review, provided graphic figures and played a key role in the thought process used throughout this paper.

1. Ambler, Scott, 2011. *Introduction to Disciplined Agile Delivery*, Retrieved February 12, 2015, from: https://www.ibm.com/developerworks/community/blogs/ambler/entry/disciplined_agile_delivery_an_introduction_white_paper22?lang=en

2. CMMI Institute – Home of the Capability Maturity Model Integration. Retrieved February 8, from: http://cmmiinstitute.com/about-cmmi-institute

3. IBM Notes. Retrieved January 13, 2015, from IBM Corporation: http://www-03.ibm.com/software/products/en/ibmnotes

4. IBM Rational Unified Process. Retrieved August 4, 2011, from IBM Corporation: https://w3-03.sso.ibm.com/services/practitionerportal/ppServlets/displayDocument.wss?syntheticKey=C076467O81896S26

5. Rational Unified Process. Best Practices for Software Development Teams. Retrieved February 10, 2015, from IBM Corporation: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

6. Software Development Methodology. Retrieved August 4, 2011, from IBM Corporation: http://en.wikipedia.org/wiki/Software_development_methodology

7. Unified Method Framework. Retrieved August 4, 2011, from IBM Corporation: http://w3-05.ibm.com/services/emea/3emgs.nsf/pages/UMFAllMaterials

8. Wells, Don, 2009. *Extreme Programming: A Gentle Introduction*, Retrieved August 5, 2011, from: http://www.extremeprogramming.org/